2-Dominant Resource Fairness: Fairness-Efficiency Tradeoffs in Multi-resource Allocation

Suhan Jiang and Jie Wu
Department of Computer and Information Sciences
Temple University

Abstract—**Fair allocation has been studied intensively in both economics and computer science. Many existing mechanisms that consider fairness of resource allocation focus on a single resource. With the advance of cloud computing that centralizes multiple types of resources under one shared platform, multi-resource allocation has come into the spotlight. In fact, fair/efficient multi-resource allocation has become a fundamental problem in any shared computer system. The widely-used solution is to partition resources into bundles that contain fixed amounts of different resources, so that multiple resources are abstracted as a single resource. However, this abstraction cannot satisfy different demands from heterogeneous users, especially on ensuring fairness among users competing for resources with different capacity limits. A promising approach to this problem is dominant resource fairness (DRF), which tries to equalize each user's dominant share (share of a user's most highly demanded resource, *i.e.*, the largest fraction of any resource that the user has required for a task), but this method may still suffer from significant loss of efficiency (*i.e.*, some resources are underused). This paper develops a new allocation mechanism based on DRF aiming to balance fairness and efficiency. We consider fairness not only in terms of a user's dominant resource, but also in another resource dimension which is secondarily desired by this user. We call this allocation mechanism 2-dominant resource fairness (2-DF). Then, we design a non-trivial on-line algorithm to find a 2-DF allocation and extend this concept to k-dominant resource fairness (k-DF).**

Index Terms—**Allocation, efficiency, fairness, multi-resource.**

I. Introduction

Resource allocation is one of the central topics in the field of computer science. There are many policies that govern resource allocation to achieve fairness among users. Max-min fairness, one of the most popular allocation policies, tries to maximize the allocation for the most poorly treated users, *i.e.*, maximize the minimum. Weighted max-min fairness adds a new concept called weight based on max-min fairness, and assigns each user with a share of the resources according to a preset weight. However, an obvious limitation in most of the existing works is that they are only devoted to single-resource allocation when quantifying the notion of fairness, *e.g.* by allocating available link bandwidth to network flows.

Fair (or efficient) multi-resource allocation is a fundamental problem in any shared computer system. A typical example is data centers that process numerous jobs with heterogeneous resource requirements on bandwidth, memory, and CPU etc. Both Hadoop and Dryad employed a simple solution based on resource abstraction. As is shown in Fig 1, all resources are partitioned into bundles with fixed amounts of different

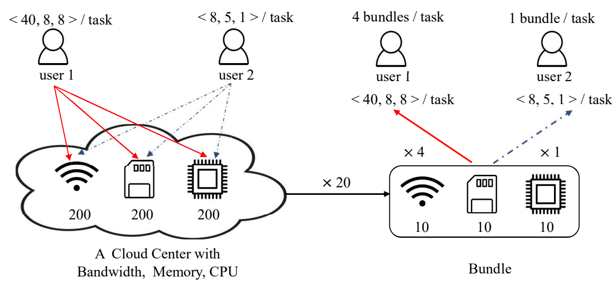


Fig. 1: A multi-resource allocation setting using resource abstraction.

resources, so that multiple resources are abstracted as a single resource. However, this simple resource abstraction ignores the different demands of heterogeneous users, and cannot always match nicely with user demands.

Ghodsi *et al.* [1] first put forward a compelling approach to this problem, which is known as dominant resource fairness (DFR). In brief, DRF allocates resources according to users' proportional demands, applying max-min fairness to each user's *dominant share*. Dominant share is the maximum share that a user has been allocated of any resource. Such a resource is then called a *dominant resource*. Although DRF has attracted much attention, this allocation approach has been questioned continuously for the reasons given below: (1) *fairness dispute* - DRF only considers one dimension when allocating all resources. Once the allocation of a user's dominant resource is determined, resources in other dimensions are proportionally assigned according to the user's request. (2) *efficiency loss* - Jin *et al.* [2] and Bertsimas *et al.* [3] respectively showed that proportional fairness, which maximizes the sum of the log of completed tasks of different users, always more efficiently uses resources than DRF does.

A. Motivation

We use the example shown in Figure 1 to illustrate existing problems in the DRF allocation mechanism. Given a system with three resources $\langle Bandwidth,\ Memory,\ CPU\rangle$ and two users $\langle user1,\ user2\rangle$, the capacity of each resource is 200 units respectively. User 1 executes each task with the request vector $\langle 40,\ 8,\ 8\rangle$, while user 2 requires $\langle 8,\ 5,\ 1\rangle$ for each task. According to DRF, user 1 gets an allocation of $\langle 100,\ 20,\ 20\rangle$, and user 2 gets $\langle 100,\ 62.5,\ 12.5\rangle$. The resulting allocation is shown in Fig. 2.

In the *fairness* domain, it is obvious that the amount of resources allocated to each user is only dependent on his

978-1-5386-6808-5/18/$31.00 © 2018 IEEE

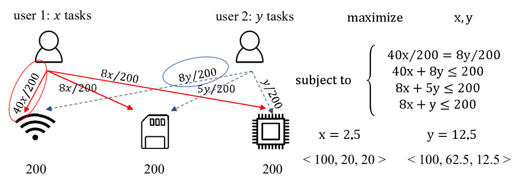


Fig. 2: An example of dominant resource fairness allocation.

dominant resource request. According to the blue line shown in Fig. 3, if user 2's request on memory (which isn't his dominant resource) varies, there is no change on the number of total tasks he can complete.

In the *efficiency* domain, if we define efficiency in terms of the aggregate tasks of all users, this example also reflects DRF's deficiency. In this setting, DRF produces an allocation with 2.5 tasks for user 1 and 12.5 tasks for user 2. This allocation brings about a significant loss in system efficiency. If we assign 2 tasks to user 1 and 15 tasks to user 2, this allocation yields 17 tasks in total, which is more efficient than DRF. Consider an extremely unequal allocation, where all resources are allocated to user 2, it will produce 25 tasks completed in total.

B. Our Result

To some extent, the previous example shows the fundamental tradeoff between fairness and efficiency: fairness and efficiency cannot be achieved simultaneously. In this paper, we seek to answer a fundamental question of resource management: how to allocate multi-type resources among users with heterogeneous demands, in an attempt to balance two opposing factors - efficiency and fairness.

Based on this question, we propose a new allocation mechanism called 2-dominant resource fairness (2-DF). A concept called 2-dominant share (denoted by s) is used. s_i is defined for each user i as the product of first two dominant demand/capacity ratios. Compared with dominant resource share, 2-dominant share is a better reflection of a user's true demand of all resources. Similarly to DRF, our allocation mechanism tries to equalize each user's 2-dominant share as much as possible. Further, we extend this 2-dominant resource fairness to k-dominant resource fairness (k-DF). We show our allocation mechanism can improve resource utility in most cases while still keeping some important fairness properties, and we also prove that our allocation mechanism satisfies both strategy-proofness and Pareto-optimality; meanwhile, we could achieve envy-freeness in some scenarios that appear quite frequently.

Our contributions in this paper are summarized as follows:

- Motivated by the deficiency of DFR policy, we reconsider the definition of fairness and efficiency, and formulate several basic metrics to measure fairness and efficiency of a specific allocation mechanism in the multi-resource scenario.
- A new allocation mechanism called 2-dominant resource fairness is proposed to fairly and efficiently allocate multiple resources to users with heterogeneous demands.

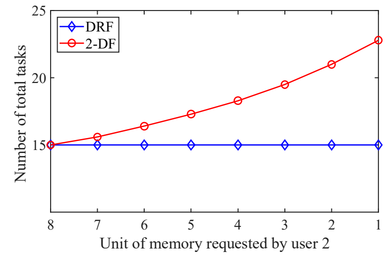


Fig. 3: User changes his request on a non-dominant resource dimension.

- An online scheduling algorithm is designed to realize 2-DF allocation mechanism.
- We prove that in 2-DF allocation mechanism, strategy-proofness and Pareto-optimality can be guaranteed, and envy-freeness can be achieved in some scenarios that appear quite frequently.
- We simulate our algorithm in real-world scenarios, and compare the performances of our algorithm with the previous approaches according to our predetermined metrics.

The remainder of the paper is organized as follows. Section II briefly reviews the related works, and Section III formulates the metrics that are to measure fairness and efficiency in multi-resource allocation settings. Section IV presents our model and related allocation mechanism. In Section V, we prove some important fairness properties achieved in our mechanism. We discuss experiment results in Section VI, and conclude our paper in Section VII.

II. Related Work

Multi-resource allocation problems arise in increasingly many applications. Data centers with multiple resources have often employed a single resource abstraction by partitioning different resources into bundles. However, multi-resource allocation viewed as single-resource allocation inevitably leads to significant inefficiencies because of the heterogeneous user demands. Different approaches have been proposed to deal with multi-resource allocation problems. Many different dimensions have been taken into account, such as desirable allocation characteristics, utility functions used to measure happiness of users, and the step at which a resource allocation approach should be applied [4].

As discussed earlier, Ghodsi *et al.* [1] proposed the DRF policy, which provides fair allocation of multiple resources in terms of dominant shares. It retains a number of desirable sharing properties and has been widely studied in both theory and practice. Then, Ghodsi *et al.* [5] extended DRF to packet networks and proposed DRFQ, the first fair multi-resource queuing algorithm. Gutman and Nisan [6] considered generalizations of DRF in a more general utility model, such as the Leontief preferences. Joe-Wong *et al.* [7] paid attention to the tradeoffs between fairness and efficiency, and generalized the DRF policy by designing a unifying multi-resource allocation framework. Parkes *et al.* [8] extended DRF in several ways, and in particular studied the case of indivisible tasks.

Wang *et al.* [9] and Friedman *et al.* [10] extended DRF's all-in-one resource model to distributed systems with heterogeneous machines, while Zeldes and Feitelson [11] proposed an on-line algorithm based on bottlenecks and global priorities. Kash *et al.* [12] extended DRF to a dynamic setting, where users dynamically arrive over time but never depart. Also, Li *et al.* [13] generalized the dynamic dominant resource fairness mechanism to the bounded case, where each user has a finite number of tasks. In their paper, a linear-time optimal algorithm is presented. Dolev *et al.* [14], on the other hand, suggested a different fairness notion for multi-resource allocation based on fairly dividing a global system bottleneck resource. Zahedi and Lee [15] applied the concept of Competitive Equilibrium from Equal Outcomes (CEEI) in the case of the CobbDouglas utilities to achieve properties similar to DRF.

Recently, a new mechanism called Greediness Metric Fairness has been developed [16], [17], [18], which can be applied to allocate physical resources by periodically adapting the priorities of virtual machines. This mechanism is highly flexible and applicable to scheduling, since it has no assumptions on utility functions. Besides, there appears a new trend to take advantage of machine learning to allocate multiple resources to satisfy user requests [19].

III. Metrics on Fairness and Efficiency

When assessing the quality of an allocation, we can distinguish two types of indicators of social welfare: fairness and efficiency. While fairness is a basic requirement of different users for resource competition, efficiency is the most desirable property for a system provider. Hence, a key challenging issue is how to balance these two factors with the desired performance and user satisfaction. Before discussing our new allocation mechanism, we start with some widely-accepted definitions of fairness and efficiency in the multi-resource allocation environments, and formulate some metrics that should be considered to measure a multi-resource allocation mechanism.

In terms of fairness, equal sharing seems to be treated as the conventional idea. However, it cannot always be a good interpretation of fairness even for a single-resource allocation. Given a total of 12-unit bandwidth and 3 network users, A needs 1.5 unit bandwidth for web-browsing, B needs 4.5 unit bandwidth for watching a film, and C needs 6 unit bandwidth to hold a video conference. Egalitarianism will lead to an allocation of 4 units for each user. 4 units for A are really a waste while for B and C are not enough. This equal allocation scheme produces a very low network resource utility. A better allocation can be $\langle 1.5,\ 4.5,\ 6\rangle$ for A, B, C, which ensures that all users can be better off or at least no worse off than in the case of equal sharing. A major challenge of multi-resource fairness is incorporating the heterogeneity of different users' requirements for different resources into the assessment of its fairness. As mentioned in [4], there are two main reasons why more complexity and less agreement are encountered in defining multi-resource fairness: one is that fairness is an intuitive concept, and the other is that the organization of resources also has influence on defining fairness. Instead of being stuck in different definitions of fairness, we list some acknowledged and important properties of a fair multi-resource allocation proposed by Ghodsi *et al.* [1].

Definition 1 (Pareto efficiency)**.** *Increasing the allocation of a user will lead to decreasing the allocation of at least another user. This means no user can run more tasks without harming someone else's benefits.*

Definition 2 (Sharing incentive)**.** *Each user should be at least no worse off by sharing resources compared with exclusively equally partitioning each resource. Given a set of resources and n users, each user should be able to run more tasks if they share resources.*

Definition 3 (Envy-freeness)**.** *A user should not prefer the allocation of another user. Changing her current allocation with that of anyone else would not improve her total task number.*

Definition 4 (Strategy-proofness)**.** *Users should not be able to benefit from lying about their resource demands, which means a user cannot run more tasks by lying.*

It is also not easy to measure efficiency in the multi-resource allocation setting. In a single-resource scenario, the most efficient allocation will clearly use the entire resources and thus achieve the maximal number of tasks. On this basis, we list two metrics to measure multi-resource allocation efficiency.

- the number of total tasks completed: higher resource efficiency is achieved by more tasks done by all users.
- the amount of unused resources: after all required resources are allocated, more unused resources allow a datacenter to serve forthcoming users.

IV. 2-Dominant Resource Fairness

In this section, we propose a new allocation mechanism called 2-dominant resource fairness (2-DF) for heterogeneous users with different requests for distinct resources.

A. Background and Model

In our model, we follow Ghodsi *et al.* [1] and assume that resources of the same type are assembled in homogeneous pools. Consider r infinitely divisible resources, the capacity of each resource j is C_j. There are n users indexed by i. Each user runs many parallel tasks. Each task is characterized by a request vector $D_i = [a_{i1}, a_{i2}, \cdots, a_{ir}]$, which specifies the amount of different resources needed during the runtime. Because the tasks from a user are typically the same binary program running on different data blocks of similar sizes, they require the same amount of resources. We therefore assume the same request vector across a user's tasks and that each user i requires an amount of each resource in fixed proportion. User i's final allocation is defined by a vector $A_i = [\varphi_i a_{i1}, \varphi_i a_{i2}, \cdots, \varphi_i a_{ir}]$, where $\varphi_i a_{ij}$ represents the fraction of resource j allocated to user i.

B. Multi-resource Fairshare Function

As discussed in the previous section, dominant share is the core part of DRF. Thus, we consider 'fairshare' which is a generalization of dominant share for each user. When allocating resources, we can easily apply max-min fairness to each user's fairshare. The drawback of dominant share is that it only considers one resource for a user and cannot allow the user to express how important this resource is in comparison with other resources.

In the following part, we propose two major rules that should be considered when defining a fairshare function in any multi-resource allocation mechanism. Then we let these rules guide us to design our allocation mechanism.

- multiple dimensions of resources: when calculating the resource allocation, a fairshare function should not consider only one dimension of all resources. It should reflect the demand of non-dominant resources as well. If two different tasks have the same demand of a given dominant resource, then the one having smaller demand of non-dominant resources should receive some compensation. Further, if a task has a totally lower demand of all given resources compared with other tasks, then it should receive more compensation.
- weights among different resources: for a given resource, a fairshare function should allow users to determine the weight to stress how important this resource is.

C. 2-Dominant Resource Fairness

In this part, we continue the previous model built on our mechanism and introduce a fairshare function called 2-dominant share in our allocation mechanism.

According to user i's request vector $D_i = [a_{i1}, a_{i2}, \cdots, a_{ir}]$, let $d_{i1} = \max\left\{\frac{a_{ij}}{C_j}\right\}$ where $j \in [1, r]$ be user i's first dominant request ratio and $d_{i2} = \max\left\{\frac{a_{ij}}{C_j}\right\} - \{d_{i1}\}$ where $j \in [1, r]$ be i's second dominant request ratio. The resources corresponding to i's dominant requests are called her dominant resources. Here, we define that each user i has a 2-dominant share expressed as $\varphi_i \cdot d_{i1} \cdot d_{i2}$.

Definition 5 (2-dominant share)**.** *The 2-dominant share of a user i is defined as.*

$$s_i = \varphi_i \cdot d_{i1} \cdot d_{i2} \tag{1}$$

In the 2-DF, we are trying to apply max-min fair allocation with respect to the users' 2-dominant share. That is, we always maximize the lowest 2-dominant share first followed by the second lowest, etc. Now, we present an example to illustrate how our 2-DF allocation mechanism works.

An Example. We still use the example from Fig. 1 to illustrate how 2-DF allocates resources according to users' different requests.

In the above scenario, each task from user 1 consumes $\frac{1}{5}$ of bandwidth, $\frac{1}{25}$ of memory and $\frac{1}{25}$ of CPU, so user 1's first and second dominant requests lie on bandwidth and memory (or CPU), respectively. Similarly, user 2's first and

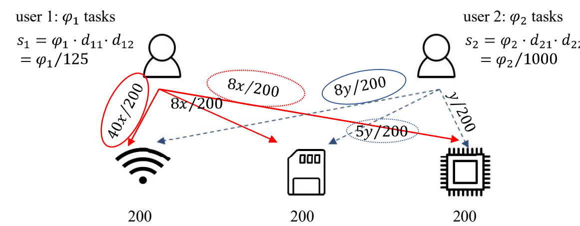


Fig. 4: Resource allocations under 2-DF.

second dominant requests lie on bandwidth and memory, respectively. 2-dominant fairness will equalize users' 2-dominant shares. The allocation can be computed mathematically as follows: Let φ_1 and φ_2 be the number of tasks allocated to user 1 and 2, respectively. Then user 1's allocation vector is $\langle 40\varphi_1,\ 8\varphi_1,\ 8\varphi_1\rangle$, and user 2's allocation vector is $\langle 8\varphi_2,\ 5\varphi_2,\ \varphi_2\rangle$. The total allocated amount of bandwidth is $(40\varphi_a + 8\varphi_b)$, the total allocated amount of memory is $(8\varphi_1 + 5\varphi_2)$, and the total allocated amount of CPU is $(8\varphi_1 + \varphi_2)$. Besides, the 2-dominant share of user 1 and user 2 is $\varphi_1 \cdot \frac{1}{5} \cdot \frac{1}{25} = \frac{1}{125} \cdot \varphi_1$, and $\varphi_2 \cdot \frac{1}{25} \cdot \frac{1}{40} = \frac{1}{1000} \cdot \varphi_2$. The 2-dominant fairness allocation is then given by the solution to the following optimization problem:

$$\begin{aligned} &\text{maximize} && \varphi_1, \varphi_2 && \text{(2a)}\\ &\text{subject to} && \frac{1}{125} \cdot \varphi_1 = \frac{1}{1000} \cdot \varphi_2 && \text{(2b)}\\ &&& 40\varphi_1 + 8\varphi_2 \le 200 &&\\ &&& 8\varphi_1 + 5\varphi_2 \le 200 &&\\ &&& 8\varphi_1 + \varphi_2 \le 200 && \end{aligned}$$

Solving this problem yields $\varphi_1 = 1.9$, and $\varphi_2 = 15.4$. Thus, user 1 gets $\langle 77,\ 15.4,\ 15.4\rangle$, and user 2 gets $\langle 123,\ 77,\ 15.4\rangle$. Recall the result mentioned in Section I-A, under DRF allocation mechanism, user 1 will receive an allocation of $\langle 100,\ 20,\ 20\rangle$, and user 2 will receive $\langle 100,\ 62,5,\ 12.5\rangle$. Thus, 2-DF leads to an increase on the number of total tasks from 15 to 17.3. Besides, if user 2's request on memory (which is his second dominant resource) varies, his completes more tasks, as the red increasing line shown in Fig. 3.

The intuition behind this allocation mechanism is, if we only consider fairness as equalizing each user's dominant share and satisfying any demand it has of other resource dimensions, it seems unfair for those users whose tasks have a low request on each resource dimension. Besides, it also leads to low efficiency because more tasks come from more resource allocations to users with smaller request vectors. Thus, we take a look at one more resource dimension to know better about a user's real request, and want to give users with lower requests on more resource dimensions more compensation to increase their total allocation shares, thus leading to more tasks done.

D. 2-DF Scheduling Algorithm

Algorithm 1 shows the pseudo-code for 2-DF scheduling. The algorithm tracks the total resources allocated to each user as well as the user's 2-dominant share, s_i. At each step, DRF picks the user with the lowest 2-dominant share among those with tasks ready to run, if that user's task demand can be

Algorithm 1: 2-DF Scheduling Algorithm

$R = [C_1, \cdots, C_r]$; # total resource capacities
$U = [u_1, \cdots, u_r]$; # total resources used by now, initially 0
$s_i\ (i = 1...n)$; # user i's 2-dominant share, initially 0
$d_{i1} = max\left\{\frac{a_{ij}}{C_j}\right\}(j = 1...r)$; # user i's first dominant request
$d_{i2} = max\left\{\frac{a_{ij}}{C_j}\right\} - \{d_{i1}\}\,(j = 1...r)$; # user i's second dominant request
$A_i = [a_{i1}, \cdots, a_{ir}]$; # resources allocated to user i, initially 0
Output 2-DF
pick user i with lowest 2-dominant share s_i;
$D_i \leftarrow$ user i's demand vector;
if $U + D_i \leq R$ **then**
    $U = U + D_i$; # update consumed vector
    $Ai = Ai + Di$; # update is allocation vector
    $s_i += d_{i1} \cdot d_{i2}$;
else
    return; # the cluster is full
end if
Output Reclaim
pick user i with one task done;
$U = U - D_i$; # update consumed vector
$U = U - D_i$; # update consumed vector
$Ai = Ai - Di$; # update is allocation vector
$s_i -= d_{i1} \cdot d_{i2}$;
$s_i += d_{i1} \cdot d_{i2}$;

satisfied, *i.e.*, there are enough resources. According to our assumption, the tasks from a user should have the same request vector, and we use variable D_i to denote the demand vector of user i. Once a launched task finishes, the user releases the task's resources and our 2-DF mechanism again selects the user with the smallest dominant share to run her task.

E. Extending from 2-D to k-D with weights

If we evaluate our 2-dominant share using the rules mentioned in Section IV-B, we could see it really takes an additional non-dominant resource into consideration. However, 2-dominant share can be extended to a better fairshare function, which considers k dimensions of resources and allows a user to stress how important each dimension is.

Let $[d_{i1}, d_{i2}, \cdots, d_{ik}]$ be the top k largest elements among $\{d_{ij}\}$ where $j \in [1, r]$, and each of the k elements is associated with a weight predefined by user i to express its importance. Then, we define that each user i has a k-dominant weighted share expressed as $\varphi_i \prod_{l=1}^{k} w_{il} \cdot d_{il}$. Similar to 2-DF, k-DF applies max-min fair allocation with respect to the users' k-dominant weighted share. That is, it always tries to equalize all users' k-dominant weighted shares. Thus, 2-dominant share is a special case of k-dominant weighted share, where $k = 2$ and $w_{i1} = w_{i2} = 1$ for each user i. For simplicity, we assume all weights are defined as 1 in the rest of this paper.

V. Properties of k-DF

Next, we will discuss some of those desirable properties satisfied by k-DF and provide intuitive explanations for our analyses. For simplicity, we normalize the capacities of r infinitely divisible resources to be 1, respectively. We begin with showing that k-DF yields Pareto-efficiency.

Theorem 1. *Every k-DF allocation is Pareto-efficient.*

Proof. Assume user i can increase her total allocation without decreasing the allocation of anyone else. In fact, every user in a k-DF allocation has at least one saturated resource. If user i is monopolizing her saturated resource, it is impossible to increase i's allocated fraction on her saturated resource. If the saturated resource is shared by user i and other users, then increasing the allocation of i must lead to decreasing the allocation of at least another user j who shares the same saturated resource, violating the hypothesis. □

Then, we show that k-DF promises Strategy-proofness.

Theorem 2. *The k-dominant fairness is Strategy-proof, i.e., any user cannot increase her allocation of every resource (only increasing fraction on some resources cannot bring about an improved task number) in the k-dominant fairness by boosting some component of her true request vector.*

Proof. If user i is monopolizing her saturated resource, it is impossible to increase i's allocated fraction on her saturated resource; thereby her total allocation cannot be increased no matter what request vector she uses. Then, we discuss the situation where user i's saturated resource is also shared by other users. Assume user i can increase her total allocation by using a different request vector $D_i{}' \neq D_i$, which means $\varphi_i{}' > \varphi_i$. Thus, user i's k-dominant share $\varphi_i{}' \cdot \prod_{l=1}^{k} d_{il}$ is increased. Since we are trying to equalize the k-dominant share of each user, any other user's k-dominant share is also increased, resulting in a larger allocation on every resource. However, in order to increase user i's total allocation, we must decrease the allocation of at least another user j sharing the same saturated resource, violating the previous analysis. □

Now, we assume there exists no completely dominant user in the 2-dominant fairness, *i.e.*, for any two users i and j, their first two dominant requests satisfy either $d_{i1} \leq d_{j1}$ and $d_{i2} \geq d_{j2}$, or $d_{i1} \geq d_{j1}$ and $d_{i2} \leq d_{j2}$.

Theorem 3. *Under the condition of no completely dominant user in the 2-dominant fairness, every 2-dominant fairness allocation is envy-free.*

Proof. Assume by contradiction that there exists envy between user i and user j. Either user i or user j can be an envier. The envied must have a strictly higher fraction of every resource that the envier wants; otherwise, the envier cannot run more tasks under its allocation. Let m and n be i's two dominant resources and p and q be j's two dominant resources such that (1) $\varphi_i a_{im} \geq \varphi_i a_{in}$; (2) $\varphi_j a_{jp} \geq \varphi_j a_{jq}$. Now, we prove the theorem based on the following two conditions.

On the first condition, user i is the envier. According to the 2-dominant resource allocation mechanism, we can get the inequality equations 4 below.

$$\text{if} \begin{cases} a_{im} \cdot a_{in} \cdot \varphi_i = a_{jp} \cdot a_{jq} \cdot \varphi_j \\ a_{im} \leq a_{jp},\ a_{in} \geq a_{jq} \end{cases} \tag{3}$$

$$\text{then} \begin{cases} \varphi_i a_{im} \leq \varphi_j a_{jp} \\ \varphi_i a_{in} \geq \varphi_j a_{jq} \end{cases} \tag{4}$$

For $\varphi_i a_{in} \geq \varphi_j a_{jq}$, there are three possible conditions:

- if n and q represent the same resource, then $\varphi_i a_{in} \geq \varphi_j a_{jn}$;
- if $a_{jq} \geq a_{jn}$, then $\varphi_i a_{in} \geq \varphi_j a_{jn}$;
- if $a_{jq} \leq a_{jn}$, meaning that resource n is user j's first dominant resource;

then $\varphi_i a_{im} \geq \varphi_i a_{in} \geq \varphi_j a_{jp} \geq \varphi_j a_{jm}$, namely $\varphi_i a_{im} \geq \varphi_j a_{jm}$. Since $\varphi_i a_{ip} \leq \varphi_i a_{im}$, then $\varphi_i a_{ip} \leq \varphi_j a_{jp}$. Thus, both user i and user j have a higher (at least equal) fraction on one resource than the other does, violating the hypothesis.

On the second condition, user j is the envier. We can get the inequalities 6 below by following the 2-dominant resource allocation mechanism.

$$\text{if} \begin{cases} a_{im} \cdot a_{in} \cdot \varphi_i = a_{jp} \cdot a_{jq} \cdot \varphi_j \\ a_{jp} \leq a_{im},\ a_{jq} \geq a_{in} \end{cases} \tag{5}$$

$$\text{then} \begin{cases} \varphi_j a_{jp} \leq \varphi_i a_{im} \\ \varphi_j a_{jq} \geq \varphi_i a_{in} \end{cases} \tag{6}$$

Quite similar to Condition 1, for $\varphi_j a_{jq} \geq \varphi_i a_{in}$, there are three possible conditions:

- if q and n represent the same resource, then $\varphi_j a_{jq} \geq \varphi_i a_{iq}$;
- if $a_{in} \geq a_{iq}$, then $\varphi_j a_{jq} \geq \varphi_i a_{iq}$;
- if $a_{in} \leq a_{iq}$, meaning that resource q is user i's first dominant resource;

then $\varphi_j a_{jp} \geq \varphi_j a_{jq} \geq \varphi_i a_{im} \geq \varphi_i a_{ip}$, namely $\varphi_j a_{jp} \geq \varphi_i a_{ip}$. Since $\varphi_j a_{jm} \leq \varphi_j a_{jp}$, then $\varphi_j a_{jm} \leq \varphi_i a_{im}$. Quite similar to the first condition, on the second condition where $\varphi_j a_{jq} \geq \varphi_i a_{in}$, we still get the same conclusion, that is, both user i and user j have a higher (at least equal) fraction on one resource than the other does, violating the hypothesis.

Under both conditions, we can achieve the same conclusion that both user i and user j have a higher (at least equal) fraction on one resource than the other does, violating the hypothesis. □

Next, we will extend this envyfree scenario from 2-d to k-d. Let $[d_{i1}, d_{i12}, \cdots, d_{i1k}]$ be the top k largest elements among $\{d_{ij}\}$ where $j \in [1, r]$. We assume there exists no completely dominant user in the k-dominant fairness, *i.e.*, for any two users i and j, their top k dominant requests satisfy:

$$\exists m, \exists n, \quad \begin{cases} \prod_{l \neq m}^{k} d_{il} \leq \prod_{l \neq m}^{k} d_{jl} \\ \prod_{l \neq n}^{k} d_{il} \geq \prod_{l \neq n}^{k} d_{jl} \end{cases} \tag{7}$$

Theorem 4. *Under the condition of having no completely dominant users in the k-dominant fairness, every k-dominant fairness allocation is envy-free.*

Proof. Assume by contradiction that there exists envy between user i and another user j. Either user i or user j can be an envier. The envied must have a strictly higher fraction of every resource that the envier wants. According to the k-dominant fairness, user i and user j should have equal k-dominant fairness, that is:

Capacity	NFC	DRF	2DF
3	1.62	1.34	1.39
5	1.81	1.48	1.56

TABLE I: Average NTT.

$$\varphi_i \cdot \prod_{l=1}^{k} d_{il} = \varphi_j \cdot \prod_{l=1}^{k} d_{jl} \tag{8a}$$

$$\text{subject to} \quad \prod_{l \neq m}^{k} d_{il} \leq \prod_{l \neq m}^{k} d_{jl} \tag{8b}$$

$$\prod_{l \neq n}^{k} d_{il} \geq \prod_{l \neq n}^{k} d_{jl} \tag{8c}$$

Then, equations in 8 can be reduced in the following form:

$$\begin{cases} \varphi_i d_{im} \geq \varphi_j d_{jm} \\ \varphi_i d_{in} \leq \varphi_j d_{jn} \end{cases} \tag{9}$$

Assume d_{im} and d_{in} represent i's requests on resource u and v, d_{jm} and d_{jn} represent j's requests on resource x and y. For $\varphi_i d_{im} \geq \varphi_j d_{jm}$, if u and x are the same resource, then $\varphi_i a_{iu} \geq \varphi_j a_{ju}$; if $a_{ju} \leq a_{jx}$, then $\varphi_i a_{iu} \geq \varphi_j a_{jx} \geq \varphi_j a_{ju}$, namely $\varphi_i a_{iu} \geq \varphi_j a_{ju}$. In addition, if $a_{ju} \geq a_{jx}$, there must exist one resource, w, which could satisfy $a_{iw} \geq a_{iu}$ && $a_{ju} \geq a_{jw}$, such that $\varphi_i a_{iw} \geq \varphi_i a_{iu} \geq \varphi_j a_{jx} \geq \varphi_j a_{jw}$, namely $\varphi_i a_{iw} \geq \varphi_j a_{jw}$. Thus, we can always find one resource of which user i has no less fraction than user j does.

For $\varphi_i d_{in} \leq \varphi_j d_{jn}$, we can also achieve a similar conclusion that there always exists one resource of which user j has no less fraction than user i does. Thus, user i has no less fraction than user j does of one resource; meanwhile, user j has no less fraction than user i does of another resource, violating the hypothesis. □

VI. PERFORMANCE EVALUATION

We consider two multi-resource allocation scenarios in different data centers to evaluate our allocation mechanism. All allocation mechanisms were implemented with MATLAB-R2017b, running on a local machine with an Intel Core 2 Duo E8400 3.0 GHz CPU and 8 GB RAM.

To measure resource efficiency, we use two metrics, which are previously mentioned in Section III. We formally define the two metrics: **NTT** which represents the number of total tasks completed by all users and **AUR** which represents the amount of unused resource after an allocation. Besides, we also compare our allocation mechanism with some existing allocation works.

A. The first scenario

In the first scenario, there is a data center with three distinct and divisible resources, r_1, r_2, and r_3. There are 3 users, each of whom requires a fixed amount of each resource to accomplish a task. Tasks are assumed to be infinitely divisible.

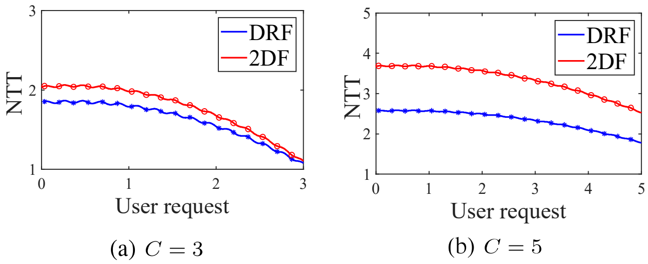


Fig. 5: Cases where higher NTT is achieved in 2-DF than DRF..

Resource capacity vector is expressed as $< C, C, C >$. We conducted two experiments with different values of C where $C \in \{3, 5\}$ to observe how resource capacity would impact fairness and efficiency. In both experiments, each user i's request vector is $< d_{i1}, d_{i2}, d_{i3} >$ where d_{ij} is an integer in the range of 1 to C for any $j = 1, 2, 3$.

1) Three comparison allocation mechanisms: We compare the efficiency in terms of **NTT** under three different allocation mechanisms: (1) No Fairness Constraints (NFC) which tries to achieve a maximal number of total tasks without considering fairness, (2) Dominant Resource Fairness (DRF) and (3) 2-Dominant Resource Fairness (2-DF).

The average **NTT** completed by 3 users with different request vectors under the three allocation mechanisms are shown in Table I. As can be seen, the number of total tasks obtained by 3 users under NFC is the highest of all allocation mechanisms. This result, to some extent, reflects the fact that fairness is often a conflicting objective against efficiency in the presence of multiple resources. Besides, 2-DF outperforms DRF, and as the resource capacity increases, 2-DF's advantage becomes more evident. Thus, we can conclude that, our allocation mechanism would have good scalability in a data center with large resource capacities.

In the first experiment, among 3^9 combinations of request vectors, 2-DF executes more tasks than DRF does in around 51.7 % of total cases. In Fig. 5(a), we show all unique cases where 2-DF performs better in terms of **NTT**. In the second experiment, the capacity was changed to 5 units for each resource. Among 5^9 combinations of request vectors, 2-DF executes more tasks than DRF does in around 58.1% of total cases. In Fig. 5(b), due to the large amount of cases, we only display those unique cases in which **NTT** obtained by 2-DF is at least 30% more than that obtained by DRF.

2) Fairness: In fact, under our 2-DF mechanism, our allocation cannot guarantee sharing incentive (SI) proposed in DRF, which proposes that each user should at least get $\frac{1}{n}$ share on its dominant resource. However, if we compare the efficiency achieved by all 2-DF allocations, which satisfy SI, with that of corresponding DRF allocations, results are still good. In the first experiment, the 2-DF allocations satisfying SI increase by around 0.04 task on average, compared with their DRF counterparts. In the second experiment, the 2-DF allocations satisfying SI increase by around 0.08 task on average, compared with their DRF counterparts.

In Section III, we mention a scenario where complete envy-freeness can be achieved. We explore the occurrence frequency of this scenario, and the result is not too bad. In the first experiment, the envyfree cases account for 64.0%, and among all cases where 2-DF obtains higher efficiency and envy-free cases occupy around 37.2%. In the second experiment, the envyfree cases account for 58.8%, and among all cases where 2-DF obtains higher efficiency, envyfree cases account for around 38.7%.

Combination	User 1	User 2
I	heavy	heavy
II	heavy	light
III	light	light

TABLE II: Three combinations of user request types

B. The second scenario

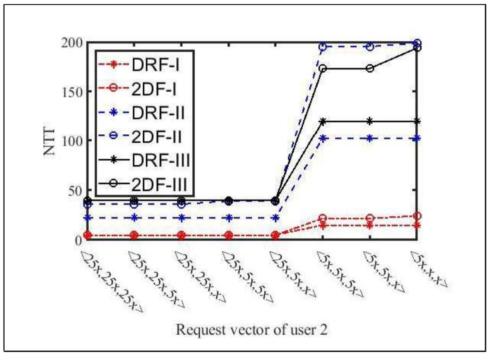
In the second scenario, we assume a data center with 1000-unit bandwidth, 1000-unit memory and 1000-unit CPU. All resources are shared by two users. Again, each user requires a fixed amount of each resource to accomplish a task, and tasks are assumed to be infinitely divisible.

In this scenario, we consider two users with different request types: heavy and light. A request $D_i =< d_{i1}, d_{i2}, d_{i3} >$ is said to be heavy, where $\forall\ j = [1, 2, 3]$, $d_{ij} \in \{25x_1, 5x_1, x_1\}$. x_1 is a random variable, which is picked randomly while following the normal distribution $\sim N(8, 0)$ in our experiments. Similarly, a request $D_i =< d_{i1}, d_{i2}, d_{i3} >$ is said to be light, where $\forall\ j = [1, 2, 3]$, $d_{ij} \in \{25x_2, 5x_2, x_2\}$ and $x_2 \sim N(1, 0)$.

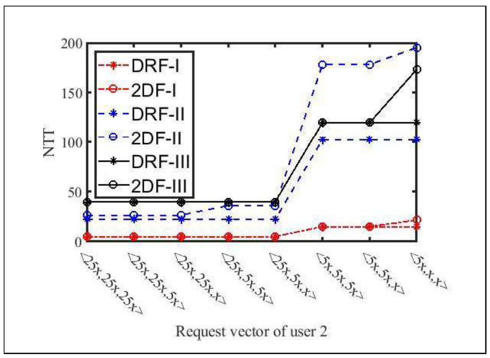
There are three combinations of user request types, as is shown in Table II. There are three request levels for both request types: big ($25x$), medium ($5x$), and small (x). Given a specific request type, we design 8 types of request vector with different request levels in different resource dimensions. We list all types here: $< 25x, 25x, 25x >$, $< 25x, 25x, 5x >$, $< 25x, 25x, x >$, $< 25x, 5x, 5x >$, $< 25x, 5x, x >$, $< 5x, 5x, 5x >$, $< 5x, 5x, x >$, and $< 5x, x, x >$. Given a specific combination of request types, there are $8 \times 8 = 64$ possible pairs of two request vectors. For each pair of request vectors, we want to compare the efficiency of DRF and 2-DF. To be precise, we conducted 10 experiments over a given pair of request vectors, by randomly choosing values of x_1 and x_2 based on their distributions Due to the large number of experiments we did, we have confidence in the consistency and reliability of the results. All results present below are the average over their own experiments.

Based on the $3 \times 8 \times 8 \times 10 = 1920$ experiments we did, we conclude two typical cases, which any other case can be induced from. We show these two cases in Fig. 6. It is obvious that, when comparing **NTT**, 2-DF outperforms DRF, especially when user 1 has big-level requests on each resource dimension and user 2 has small-level requests on each resource dimension, whatever their request types are. The average increase is 45% when comparing **NTT** of 2-DF and DRF among all experiments.

To measure efficiency of DRF and 2-DF in terms of **AUR**, we calculate the average amount of unused resources under



(a)
I: user 1 $\{25x_1, 25x_1, 25x_1\}$
II: user 1 $\{25x_1, 25x_1, 25x_1\}$
III: user 1$\{25x_2, 25x_2, 25x_2\}$



(b)
I: user 1 $\{25x_1, 5x_1, x_1\}$
II: user 1 $\{25x_1, 5x_1, x_1\}$
III: user 1$\{25x_2, 5x_2, x_2\}$

Fig. 6: Two typical cases.

Ave AUR	DRF	2DF
I	418	654
II	418	802
III	418	654

(a)
I: user 1 $\{25x_1, 25x_1, 25x_1\}$
II: user 1 $\{25x_1, 25x_1, 25x_1\}$
III: user 1$\{25x_2, 25x_2, 25x_2\}$

Ave AUR	DRF	2DF
I	1218	1271
II	1218	998
III	1218	1271

(b)
I: user 1 $\{25x_1, 5x_1, x_1\}$
II: user 1 $\{25x_1, 5x_1, x_1\}$
III: user 1$\{25x_2, 5x_2, x_2\}$

TABLE III: Average AUR .

DRF and 2-DF, respectively in two cases. The result is shown in Table III. It is obvious that 2-DF consumes less resources while yielding more tasks, which allows a data center to serve more users in the future.

VII. Conclusion

In this paper, we consider the open problem of multi-resource sharing for heterogeneous users. We show that DRF suffers from serious fairness concerns without utility guarantees. We try to mitigate this problem with the new sharing policy known as 2-dominant resource fairness, and we extend this concept to k-dominant resource fairness. In our mechanism, 2-dominant share is defined for each user as the product of first two dominant demand/capacity ratios, and we try to equalize the 2-dominant share of all users as much as possible. 2-dominant resource fairness provides strategy-proofness, in that no user can run more tasks by lying about its demands. Meanwhile, this policy is Pareto-optimal, and envy-freeness can be achieved in certain scenarios. Besides, in the traditional sense of fairness, 2-DF considers more resource dimension when allocating, and results in a more balanced allocation between dominant and non-dominant resources. Compared with dominant resource fairness (DRF), our proposed model achieves better efficiency, in terms of the number of total tasks completed by all users. It is vital for efficiency-needed applications, allowing a small penalty on widely-accepted fairness properties.

VIII. Acknowledgment

This research was supported in part by NSF grants CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and IIP 1439672.

References

[1] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *NSDI*, 2011.

[2] Y. Jin and M. Hayashi, "Efficiency comparison between proportional fairness and dominant resource fairness with two different type resources," in *CISS*, 2016.

[3] D. Bertsimas, V. F. Farias, and N. Trichakis, "The price of fairness," in *Operational Research, vol. 59, no. 1, pp. 1731*, 2011.

[4] P. Poullie, T. Bocek, and B. Stiller, "A survey of the state-of-the-art in fair multi-resource allocations for data centers," in *IEEE Transactions on Network and Service Management*, 2017.

[5] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica, "Multi-resource fair queuing for packet processing," in *SIGCOMM*, 2012.

[6] A. Gutman and N. Nisan, "Fair allocation without trade," in *Proc. AAMAS*, 2012.

[7] C. Wong, S. Sen, T. Lan, and M. Chiang, "Multi-resource allocation: fairness efficiency tradeoffs in a unifying framework," in *IEEE/ACM Transactions on Networking*, 2013.

[8] D. Parkes, A. Procaccia, and N. Shah, "Beyond dominant resource fairness: extensions, limitations, and indivisibilities," in *Proc. ACM EC*, 2012.

[9] W. Wang, B. Liang, and B. Li., "Multi-resource fair allocation in heterogeneous cloud computing systems," in *IEEE Trans. Parallel Distrib.Syst*, 2015.

[10] E. Friedman, A. Ghodsi, and C. Psomas, "Strategyproof allocation of discrete jobs on multiple machines," in *Proc. ACM EC*, 2014.

[11] Y. Zeldes and D. G. Feitelson, "On-line fair allocations based on bottlenecks and global priorities," in *Proc. ACM/SPEC ICPE*, 2013.

[12] I. Kash, A. Procaccia, and N. Shah, "No agent left behind: Dynamic fair division of multiple resources," in *Proc. AAMAS*, 2013.

[13] W. Li, X. Liu, and X. Zhang, "A further analysis of the dynamic dominant resource fairness mechanism," in *Xiao, M., Rosamond, F. (eds.) FAW*, 2017.

[14] D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial, "No justified complaints: On fair sharing of multiple resources," in *Proc. ACM ITCS*, 2012.

[15] S. Zahedi and B. Lee, "Ref: resource elasticity fairness with sharing incentives for multiprocessors," in *Proc. ASPLOS*, 2014.

[16] P. Poullie and B. Stiller, "Cloud flat rates enabled via fair multi-resource consumption," in *Proc. AIMS*, 2016.

[17] P. Poullie, S. Mannhart, and B. Stiller, "Virtual machine priority adaption to enforce fairness among cloud users," in *Proc. CNSM*, 2016.

[18] P. Poullie and B. Stiller, "The design and evaluation of a heaviness metric for cloud fairness and correct virtual machine configurations," in *Proc. GECON*, 2016.

[19] W. Chen, Y. Xu, and X. Wu, "Deep reinforcement learning for multi-resource multi-machine job scheduling," in *ArXiv e-prints,1711.07440*, 2017.